UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/073,669	02/11/2002	Kenton E. Noble	50277-1955	4587

29989 7590 01/27/2005

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 01/27/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Offic Action Summary	Applicati n N .	Applicant(s)	
	10/073,669	NOBLE ET AL.	
	Examiner	Art Unit	
	Mary J. Steelman	2122	

-- The MAILING DATE of this communication appears on the c ver sheet with th corresp ndence address --

Period f r Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *2/11/02,8/12/02*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disp sition of Claims

4)☒ Claim(s) *1-63* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-63* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *11 February 2002* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Pri rity under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing R view (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-63 are pending.

Specification

2. The use of the trademark JAVA has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

Drawings

3. The use of the trademark JAVA has been noted in the drawings. It should be capitalized wherever it appears and be accompanied by the generic terminology.

4. New corrected drawings in compliance with 37 CFR 1.121(d). Applicant is advised to employ the services of a competent patent draftsperson outside the Office, as the U.S. Patent and Trademark Office no longer prepares new drawings. The corrected drawings are required in reply to the Office action to avoid abandonment of the application. The requirement for corrected drawings will not be held in abeyance.

Claim Objections

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person

having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-63 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent 6,535,894 B1 to Schmidt et al.

Per claims 1, 19, and 37:

Schmidt disclosed:

-system, method and computer readable medium...providing a Java code release infrastructure with granular code patching, comprising:

Col. 19, line 3 – col. 20, line 65, "method", "system", "program storage device readable by a machine, tangibly embodying a program..(computer readable medium)", col. 1, lines 12-13, "facilitate incremental updating of program code(granular code patching)."

-one or more Java code patches, each comprising at least one resource unit,

Col. 5, lines 44-48, "A target archive file (code modified from the original code)....result from making changes to the program code...would typically represent the resources required to execute an updated version...", col. 5, lines 61-63, "A difference archive file is a file that describes the changes between an original archive file and a new (target) archive file", the difference archive file (a patch file) (col. 10, lines 54-59) "contains all of the entries that are new in the target archive file...along with a set of instructions for creating a synthesized archive file (a patched file) based only on the original archive file and the difference archive file.

-each resource unit comprising metadata and file components;

Col. 5, line 65, "comprises an index file describing the changes (metadata)...and also comprises a set of entries (file components) corresponding to the entries in the target archive file that are not contained in the original archive file."

-one or more Java code libraries, each comprising at least one such resource unit;
FIG. 4, #410, JAVA Archive File, col. 5, lines 25-34, "An archive file (JAVA code) is any file organized into a set of one or more entries, where each entry is itself a file...each entry in an archive file may itself be an archive file (There may be a collection of JAR files in a JAR code library)...A JAR file is one type of archive file."

-a patch tool, comprising:
-a compare module comparing the metadata for each such resource unit in the Java code patches to the metadata for each such corresponding resource unit in the Java code libraries;
Col. 17, lines 10-13, "technique which...allows a that describes the change between two JAR files to be computed (compare resources)."
-a merge module merging each such resource unit in the Java code patches into the Java code libraries for each such corresponding resource unit that is out-of-date.
Col. 18, line 24, "apply a jardiff to an original JAR file..."

Schmidt disclosed incremental updating of program code (col. 1, lines 11-12) by creating a difference file that only includes the changed or added class files contained in a JAR file (col. 17, lines 62-64). While Schmidt failed to explicitly show 'JAVA code libraries', he did disclose that

JAR files may contain JAR files. In other words a JAR file may be a library of JAR files, each of which may be processed to create a patch (difference file). Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Schmidt's invention to disclose that JAVA code libraries comprising at least one such resource unit (individual JAR files or individual class files or any file needed by the application) were available.

Per claims 2 and 20:

-an extract module extracting at least one resource unit from the Java code libraries and modifying one or more Java archive files that are out-of-date with the at least one extracted resource unit.

Col. 17, lines 44-50, "the present invention provides a tailored diff format for JAR files...it uses knowledge about the underlying structure of archive files such as JAR files, in order to provide a simple generation algorithm which guarantees that only modified or new entries are transferred..." An algorithm extracts resource units from the JAVA code library to provide a patch for any changed class or any new class added (col. 17, lines 17-18).

Per claims 3 and 21:

-a sign module signing the Java archive files using a digital certificate.

Col. 17, lines 22-23, "a jardiff file can be signed, using standard JAR signing tools."

Per claims 4 and 22:

-the one or more Java archive files are modified through at least one of creation, revision or deletion.

Col. 17, lines 17-18, "For each changed entry (e.g., a class) (revision) or new entry (creation) in new.jar (relative to org.jar) there will be an entry in jardiff file. A change in a class or a new class file added will be reflected in a difference file and used to modify the original JAR files to form a new JAR file.

Per claims 5 and 23:

-a source repository storing the source file components;

FIG. 4, #410, JAVA Archive File, col. 5, lines 25-34, "An archive file (JAVA code libraries) is any file organized into a set of one or more entries (source file components), where each entry is itself a file...each entry in an archive file may itself be an archive file (There may be a collection of JAR files in a JAR code library)...A JAR file is one type of archive file." Col. 11, lines 23-28, "FIG. 8 depicts a block diagram illustrating the file names and file contents of the entries in an exemplary original archive file (source repository) ...each entry has a unique file name..."

-a staged patch repository storing the one or more Java code patches;

Col. 11, line 65-col. 12, line 3, "entries in the target archive file (patch repository)...As would be expected in the case of a version update to program code..."

-a staged code repository organizing the one or more Java code libraries and the Java archive files.

FIG. 8 shows original code organized / stored in a repository, and updated code organized / stored in a repository.

Per claims 6 and 24:

-a resource unit generator processing the file components into at least one such resource unit;

-a packager packaging at least one such resource unit into one or more of the Java code patches.

Col. 3, lines 56-62, "a difference archive file is created (process file components into resource unit)...comprises an index file describing the changes between the original archive file and the target archive file, and also comprises a set of entries corresponding to the entries in the target archive file that are not contained in the original archive file." At the generating / packaging computer site, a difference file is created to provide a patch and prepared for transmission to a client computer.

Per claims 7 and 25:

-stored Java source code provided as the file components.

Col. 6, lines 52-54, "Code to implement the present invention may be operably disposed in system memory...", col. 5, lines 25-29, "An archive file is any file organized...may comprise any file type...application code (source code)..."

Per claims 8 and 26:

-a compiler compiling at least one Java source code file into one or more Java classes;

Col. 6, lines 52-54, "Code to implement the present invention may be operably disposed in system memory...", col. 5, lines 25-29, "An archive file is any file organized...may comprise any file type...class files..."

While Schmidt failed to provide specific details on compiling JAVA code into class code, it is well known that the code must be compiled into class files before it is executed, thus it would be obvious and inherent.

-a resource unit packager module storing the Java classes into at least one such resource unit as the file components.

Col. 5, lines 25-34, "An archive file is any file organized into a set of one or more entries, where each entry is itself a file...A JAR file is one type of archive file...", col. 17, lines 9-14, "the present invention describes a technique which...allows a file that describes the change between tow JAR files to be computed (resource unit packager)." The computed file stores JAVA classes (file components) into resource unit, using a "simple algorithm (col. 17, lines 19-20)."

Per claims 9 and 27:

-at least one of non-Java source and derived code provided as the file components.

Col. 10, line 1, "A jardiff file contains the following set of entries:", col. 18, lines 7-9, "it contains an index file, META-INF/jardiff, that describes the contents of the to.jar file, and how it relates to the from.jar file.", col. 18, line 12, "The index file describes ho to transform..."

Per claims 10 and 28:

-third party code provided as the file components.

Col. 5, lines 25-34, "An archive file is any file organized into a set of one or more entries, where each entry is itself a file. Each entry in the archive file may comprise any file type (third party code) …", col. 8, lines 55-67, "a JAR archive file typically contains a manifest file named META...within the archive file. This file contains information about the other files (third party code) within the JAR file. Application that work with JAR files need to access the information contained in the manifest file…The MANIFEST…file contains arbitrary information about the files in the archive, such as their encoding or language…"

Per claims 11 and 29:

-a metadata generator generating the metadata for each such resource unit;

Col. 8, lines 55-56, ""a JAR archive file typically contains a manifest file named META…within the archive file. This file contains information about the other files (third party code) within the JAR file."

-a resource unit packager module storing the generated metadata into the resource unit.

Col. 17, lines 19-20, "the present invention provides a simple algorithm for computing and applying (resource unit packager, stores generated metadata) the jardiff file…"

Per claims 12 and 30:

-the metadata comprises at least one of a unique identifier and a version attribute.

Naming conventions are practiced to uniquely identify classes and versions. As an example see col. 11, lines 22-39. Col. 11, lines 54-55, "the file name of each entry should be unique..."

Per claims 13 and 31:

-a compare module using a set of rules allowing one of an older resource unit to be replaced by a newer resource unit and a newer resource unit to be replaced by an older resource unit to back out a previously-applied Java code patch.

Col. 16, lines 31-52, "applying a difference archive file...the new entries in the difference archive file are copied...entries specified by copy commands in the index file...are copied...depending on the arguments specified...if any delete commands are included in the index file...The order in which the steps are performed is not critical..." The index file has commands (rules) that are executed that determine which files are replaced.

Per claims 14 and 32:

-one or more Java archive files, each comprising at least one resource unit corresponding to one such resource unit in the Java code libraries;

FIG. 4, #410, JAVA Archive File, col. 5, lines 25-34, "An archive file (JAVA code libraries) is any file organized into a set of one or more entries, where each entry is itself a file...each entry in an archive file may itself be an archive file (There may be a collection of JAR files in a JAR code library)...A JAR file is one type of archive file", col. 16, lines 60-61, "each entry in an archive file may itself be an archive file..."

-a patch tool referencing Java archive file definitions which each correspond to one or more of the Java archive files.

Col. 3, lines 57-58, "a difference archive file (patch tool) is created. The difference archive file comprises an index file (referencing definitions) describing the changes...and also comprises a set of entries (individual files or nested JAR files) corresponding to the entries in the target archive file (desired updated code file) that are not contained in the original archive file (original code file)."

Per claims 15 and 33:

-an extract module extracting the resource units from the Java code libraries into the Java archive files for each such corresponding resource unit that is out-of-date.

Col. 17, lines 19-20, "the present invention provides a simple algorithm for computing (extracting the resource units to patch corresponding out of date resource units) and applying the jardiff file..."

Per claims 16 and 34:

-an extract module referencing third party Java code libraries not maintained as part of the infrastructure.

Col. 5, lines 25-34, "An archive file is any file organized into a set of one or more entries, where each entry is itself a file. Each entry in the archive file may comprise any file type (third party code) ...", col. 8, lines 55-67, "a JAR archive file typically contains a manifest file named META...within the archive file. This file contains information about the other files (third party

code) within the JAR file. Application that work with JAR files need to access the information contained in the manifest file...The MANIFEST...file contains arbitrary information about the files in the archive, such as their encoding or language..." Schmidt suggests that any type of code (third party JAVA code libraries) may be included in the JAR archives. Schmidt also suggests that code may be retrieved from www (col. 7, lines 1-2).

Per claims 17 and 35:

-Java code libraries implemented as a portable virtual file system which can be used directly by a Java Virtual Machine.

Col. 7, lines 21-22, "browser is JAVA enabled, meaning that it includes at least one version of the JAVA Runtime Environment (includes JAVA virtual machine)", col. 7, lines 44-47, "When JAVA enabled browser encounters a link (also known as a 'tag')...it is able to download the specified applet (portable virtual file system used directly by JAVA virtual machine)..."

Per claims 18 and 36:

-a machine portable infrastructure providing support for Java language features by encapsulating Java inner classes, nested directory structures, native class names, and native character set.

FIG. 4, #410, JAVA Archive File, col. 5, lines 25-34, "An archive file is **any** file organized into a set of one or more entries, where each entry is itself a file...each entry in an archive file may itself be an archive file...A JAR file is one type of archive file." (emphasis added) JAVA inherently is portable. JAVA by definition allows inner classes, nested directory structures, native class names, and native character set. Although Schmidt failed to provide these explicit

details, the definition of the JAVA programming language includes these features. Thus this would be obvious.

Per claims 38, 42, and 46:

-system, method, computer readable media...for patching staged code in a staged Java code release infrastructure, comprising:

Col. 1, lines 12-13, "apparatus (system) and method to facilitate incremental updating of program code...", col. 6, lines 52-54, "Code to implement the present invention may be operably... stored on storage media (computer readable media)..."

-a staged code repository maintaining one or more staged Java code libraries, each staged Java code library comprising at least one resource unit, each resource unit comprising metadata and file components;

Original archive file and target archive file are code libraries of the initial code and the updated code respectively. See FIG. 7, #710 & #720.

-a staged patch repository storing one or more Java code patches, each Java code patch comprising at least one resource unit corresponding to one such resource unit specified in a Java code patch definition;

A patch repository is created using a diff algorithm. The patch comprises the files (resources) required to apply the patch. See FIG. 7, #730.

-a patch tool accessing one or more Java code patches in the staged patch repository, comprising:

-a compare module comparing the metadata for each resource unit in the Java code patches to the metadata in the staged Java code libraries for each such corresponding resource unit;

Col. 17, lines 16-18, "the algorithm woks on the level of entries (class files) in a JAR file. For each changed entry (e.g., class) or new entry...there will be an entry in the jardiff file...the present invention provides a simple algorithm for computing and applying the jardiff file..."

-a merge module merging each resource unit in the Java code patches into the staged Java code libraries for each such corresponding resource unit that is out-of-date.

Col. 17, lines 55-59, "The jardiff format describes how to apply incremental updates to a JAR file. For example, given two JAR files...a jardiff file can be computed...The jardiff file can be applied (merge resource unit) to org.jar (original code that is out of date) to yield targetprime.jar (incrementally patched / updated code)."

Additionally, see rejection of limitations as addressed in claim 1 above.

Schmidt disclosed incremental updating of program code (col. 1, lines 11-12) by creating a difference file that only includes the changed or added class files contained in a JAR file (col. 17, lines 62-64). While Schmidt failed to explicitly show 'JAVA code libraries', he did disclose that JAR files may contain JAR files. In other words a JAR file may be a library of JAR files, each of which may be processed to create a patch (difference file). Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Schmidt's invention to disclose that JAVA code libraries comprising at least one such resource unit

(individual JAR files or individual class files or any file needed by the application) were available.

Per claims 39 and 43:

-an extract module referencing Java archive file definitions which each correspond to a staged Java archive file, each staged Java archive file comprising at least one resource unit corresponding to one such resource unit in the staged Java code libraries.

See rejection of limitations as addressed in claim 2 above.

Per claims 40 and 44:

-an extract module extracting one such resource unit from the staged Java code libraries into the staged Java archive files for each such corresponding resource unit that is out-of-date.

See rejection of limitations as addressed in claim 2 above.

Per claims 41 and 45:

-a sign module creating a digital signature for the staged Java archive files using a digital certificate.

See rejection of limitations as addressed in claim 3 above.

Per claims 47, 55, and 63:

-system, method, computer readable media for generating Java code patches in a Java code release infrastructure, comprising:

Col. 1, lines 12-13, "apparatus (system) and method to facilitate incremental updating of program code...", col. 6, lines 52-54, "Code to implement the present invention may be operably... stored on storage media (computer readable media)..."

-a source code repository maintaining one or more source files;

Col. 17, lines 41-43, "updates can be performed either 'in place', or new JAR files can be created based on old versions, keeping the original version intact..."

-a patch generator generating one or more Java code patches, each comprising at least one resource unit, each resource unit comprising metadata and file components specified in Java code patch definitions.

FIGs. 9-11

Additionally, see limitations addressed in claims 1 and 5 above.

Schmidt disclosed incremental updating of program code (col. 1, lines 11-12) by creating a difference file that only includes the changed or added class files contained in a JAR file (col. 17, lines 62-64). While Schmidt failed to explicitly show 'JAVA code libraries', he did disclose that JAR files may contain JAR files. In other words a JAR file may be a library of JAR files, each of which may be processed to create a patch (difference file). Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify Schmidt's invention to disclose that JAVA code libraries comprising at least one such resource unit (individual JAR files or individual class files or any file needed by the application) were available.

Per claims 48 and 56:

-a resource unit generator processing the file components into at least one such resource unit;

-a packager packaging at least one resource unit into one or more of the Java code patches.

See rejection of limitations addressed in claim 6 above.

Per claims 49 and 57 :

-one or more Java source code files provided as the file components.

See rejection of limitations addressed in claim 7 above.

Per claims 50 and 58:

-a compiler compiling the Java source code into one or more Java classes;

-a resource unit packager module storing the Java classes into at least one such resource unit as file components.

See rejection of limitations addressed in claim 8 above.

Per claims 51 and 59:

-at least one of non-Java source and derived code provided as the file components.

See rejection of limitations addressed in claim 9 above.

Per claims 52 and 60:

-staged third party code provided as the file components.

See rejection of limitations addressed in claim 10 above.

Per claims 53 and 61:

-a metadata generator generating the metadata for each such resource unit;

-a resource unit packager module storing the generated metadata into one such resource unit.

See rejection of limitations addressed in claim 11 above.

Per claims 54 and 62:

-the metadata comprises at least one of a unique identifier and a version attribute.

See rejection of limitations addressed in claim 12 above.

Conclusion

7. Also note: Similarly, the IBM technical bulletin, "System for Service Level Identification in a Client / Server WWW / JAVA Environment" (1998) (hereafter referred to as "System for Service"), disclosed (last two lines of page 1) "granular fix packages which contain only the product components that have changed." System for Service disclosed. (page 2, 2nd half), "intermediate fixes (patch tool) , JAR files, different components in a JAVA environment: JAVA source, HTML, help files, OEM (third party code). Page 3, middle of the page, "These serviceable units would be the most granular level of a fix (patches made by only replacing portions of the contents/resources of a JAR file) that would be provided to a customer..." "Each serviceable unit will have a service level associated with it. A method will be provided to

determine the service level (a compare module comparing the metadata)...The service level information will be updated on each build..." Bottom of page 3, "The CAB and JAR files provide a secure package by being signed (a sign module / digital certificate)..." Middle of page 4, "...properties files that defines the product name and version/release/modification (unique identifier and version attribute)..." Bottom of page 4, "All properties files (source code) are sent through a bundling (packager) utility to convert them (resource unit generator) into a .java file (compiled .java code), which are then compiled into a resource file (class code). The generated client service level information files are packaged (packager) in the corresponding client CAB and JAR files." The properties files that list the client and server serviceable units are also shipped with the product, so that they can later be used to get the real-time service level information (meta data). Top of page 5, "There is a class that can be instantiated that will go out and read the service level information for each client...and return the information..." The IBM technical disclosure also broadly suggests the claim limitations.

8. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Mary Steelman, whose telephone number is (571) 272-3704. The examiner can normally be reached Monday through Thursday, from 7:00 AM to 5:30 PM If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached at (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Mary Steelman

01/13/2005

WEI Y. ZHEN
PRIMARY EXAMINER